ECON 950 — Winter 2020

Prof. James MacKinnon

4. Linear Methods for Classification

The output variable is now discrete. We want to divide up the space of the input variables into a collection of regions associated with K different predicted outcomes (or groups, or classes).

Let $\boldsymbol{Y}$ be a matrix with K columns of observations on the output variables. Each column contains 0s and 1s, and each row contains a single 1.

For example, if observation 44 belongs to group 3, row 44 of $\boldsymbol{Y}$ will have a 1 in column 3 and 0s in all other columns.

The OLS estimator is

$$\hat{\boldsymbol{B}} = (\boldsymbol{X}^\top \boldsymbol{X})^{-1} \boldsymbol{X}^\top \boldsymbol{Y}, \tag{1}$$

where $\hat{\boldsymbol{B}}$ is $(p+1) \times K$ and $\boldsymbol{X}$ has $p+1$ columns, one of them a constant term.

This is a generalization of the **linear probability model**. The latter is used when there are only two outcomes. In that case, we can use just one regression, because the fitted values must sum to 1 over all the equations.

In general, we could get away with estimating $K-1$ regressions and obtaining the fitted values for the K^{th} by using this property.

For any input vector $\boldsymbol{x}$, we can calculate the fitted output $\hat{\boldsymbol{f}}(\boldsymbol{x}) = [1, \boldsymbol{x}^\top]\hat{\boldsymbol{B}}$, which is a K-vector.

Then we classify $\boldsymbol{x}$ as belonging to group (or class, or region) k if k corresponds to the largest element of $\hat{\boldsymbol{f}}(\boldsymbol{x})$.

Unfortunately, as ESL explains, linear regression often performs very badly when $K \geq 3$. The problem is "masking," where middle classes get missed.

ESL works through an example where the largest value of $\boldsymbol{x}^\top\hat{\boldsymbol{B}}_k$ is always for one of the two extreme classes, even though it is easy to see visually that there are three classes which can be separated without error.

This example is shown in ESL-fig4.02.pdf.

4.1. Linear Discriminant Analysis

Suppose that $f_k(\boldsymbol{x})$ is the density of $\boldsymbol{X}$ in class k, where k runs from 1 to K.

Suppose that π_k is the prior probability of class k, where the event $G = k$ means that the class actually is k.

Then, by Bayes' theorem,

$$\Pr(G = k \,|\, \boldsymbol{x}) = \frac{\pi_k f_k(\boldsymbol{x})}{\sum_{\ell=1}^{K} \pi_\ell f_\ell(\boldsymbol{x})}. \tag{2}$$

So we just need to find the $f_k(\boldsymbol{x})$ and combine them with the prior probabilities.

If the density of each class is multivariate normal,

$$f_k(\boldsymbol{x}) = \frac{1}{(2\pi)^{p/2}|\boldsymbol{\Sigma}_k|^{1/2}} \exp\bigl(-\tfrac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu}_k)^\top \boldsymbol{\Sigma}_k^{-1}(\boldsymbol{x} - \boldsymbol{\mu}_k)\bigr). \tag{3}$$

In the case of **linear discriminant analysis**, we assume that $\boldsymbol{\Sigma}_k = \boldsymbol{\Sigma}$ for all k.

In general, the log of the ratio of the posteriors (that is, the log odds) is

$$\log \frac{\pi_k}{\pi_\ell} + \log \frac{f_k(\boldsymbol{x})}{f_\ell(\boldsymbol{x})}. \tag{4}$$

When the densities are given by (3) with constant $\boldsymbol{\Sigma}$, this reduces to

$$\log \frac{\pi_k}{\pi_\ell} - \frac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu}_k)^\top \boldsymbol{\Sigma}^{-1}(\boldsymbol{x} - \boldsymbol{\mu}_k) + \frac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu}_\ell)^\top \boldsymbol{\Sigma}^{-1}(\boldsymbol{x} - \boldsymbol{\mu}_\ell). \tag{5}$$

Because the two covariance matrices are the same, this simplifies to

$$\log \frac{\pi_k}{\pi_\ell} - \frac{1}{2}(\boldsymbol{\mu}_k - \boldsymbol{\mu}_\ell)^\top \boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_k - \boldsymbol{\mu}_\ell) + \boldsymbol{x}^\top \boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_k - \boldsymbol{\mu}_\ell), \tag{6}$$

which is linear in $\boldsymbol{x}$.

Since this is true for any pair of classes, all boundaries must be hyperplanes.

Where else have we seen a model in which the log of the odds is linear in $\boldsymbol{x}$? The **logistic regression** or **logit** model!

The **linear discriminant function** for class k is

$$\delta_k(\boldsymbol{x}) = \boldsymbol{x}^\top \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_k - \frac{1}{2} \boldsymbol{\mu}_k^\top \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_k + \log \pi_k. \tag{7}$$

We simply classify $\boldsymbol{x}$ as belonging to the class k for which (7) is largest.

Of course, for all the classes, we need to estimate

$$\hat{\pi}_k = N_k / N, \tag{8}$$

$$\hat{\boldsymbol{\mu}}_k = \frac{1}{N_k} \sum_{i \in G_k} \boldsymbol{x}_i \tag{9}$$

$$\hat{\boldsymbol{\Sigma}} = \frac{1}{N - K} \sum_{k=1}^{K} \sum_{i \in G_k} (\boldsymbol{x}_i - \hat{\boldsymbol{\mu}}_k)(\boldsymbol{x}_i - \hat{\boldsymbol{\mu}}_k)^\top. \tag{10}$$

Here G_k is the set of observations that belong to class k.

Since the values of $\delta_k(\boldsymbol{x})$ depend on the $\hat{\pi}_k$, we could change the boundaries of the classes by using different estimates of the π_k.

For example, we could shrink them towards $1/K$:

$$\hat{\pi}_k(\alpha) = \alpha \frac{N_k}{N} + (1 - \alpha)\frac{1}{K}. \tag{11}$$

When N is small and the N_k are not too different, this ought to produce better results by accepting more bias in return for less variance.

ESL discusses the relationship between LDA and classification by linear regression. For two classes, there is a close relationship.

4.2. Quadratic Discriminant Analysis

If the $\boldsymbol{\Sigma}_k$ matrices are not equal, then the log odds does not simplify to (6).

In contrast to (7), the discriminant functions are now quadratic in $\boldsymbol{x}$. The **quadratic discriminant functions** are

$$\delta_k(\boldsymbol{x}) = -\frac{1}{2}\log|\boldsymbol{\Sigma}_k| - \frac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu}_k)^\top \boldsymbol{\Sigma}_k^{-1}(\boldsymbol{x} - \boldsymbol{\mu}_k) + \log \pi_k. \tag{12}$$

Now we have to estimate separate covariance matrices for each class:

$$\hat{\boldsymbol{\Sigma}}_k = \frac{1}{N_k} \sum_{i \in G_k} (\boldsymbol{x}_i - \hat{\boldsymbol{\mu}}_k)(\boldsymbol{x}_i - \hat{\boldsymbol{\mu}}_k)^\top. \tag{13}$$

Note that these matrices are $p \times p$, each with $p(p+1)/2$ parameters to estimate.

Especially when some of the N_k are small, the $\boldsymbol{\Sigma}_k$ may not be estimated very well, which will probably cause the classification procedure to perform much worse on the test data than on the training data.

There is an interesting alternative to QDA that is based on LDA.

Simply augment $\boldsymbol{x}$ by adding all squares and cross-products, and then perform LDA. See ESL-fig4.01.pdf and ESL-fig4.06.pdf. Of course, this also adds a lot of parameters if p is large.

ESL reports that LDA and QDA often work remarkably well, even though the Gaussian assumption, and the equal covariance matrix assumption, are surely not true in the vast majority of cases.

Presumably, the parsimony of LDA causes it to have low variance but high bias. In many cases, it is apparently worth accepting a lot of bias in return for low variance.

Perhaps the data can only support simple decision boundaries such as linear or quadratic ones, and the estimates provided via the Gaussian LDA and QDA models are stable.

Since we need $|\hat{\boldsymbol{\Sigma}}_k|$ and $\hat{\boldsymbol{\Sigma}}_k^{-1}$ rather than just $\hat{\boldsymbol{\Sigma}}$, it is convenient to use the eigen decomposition

$$\hat{\boldsymbol{\Sigma}}_k = \boldsymbol{U}_k \boldsymbol{D}_k \boldsymbol{U}_k^\top. \tag{14}$$

Then

$$\log |\hat{\boldsymbol{\Sigma}}_k| = \sum_{\ell=1}^{p} \log d_{k\ell}, \tag{15}$$

and

$$(\boldsymbol{x} - \hat{\boldsymbol{\mu}}_k)^\top \hat{\boldsymbol{\Sigma}}_k^{-1} (\boldsymbol{x} - \hat{\boldsymbol{\mu}}_k) = \big(\boldsymbol{U}_k^\top (\boldsymbol{x} - \hat{\boldsymbol{\mu}}_k)\big)^\top \boldsymbol{D}_k^{-1} \big(\boldsymbol{U}_k^\top (\boldsymbol{x} - \hat{\boldsymbol{\mu}}_k)\big). \tag{16}$$

Getting the eigenvalues and eigenvectors is expensive, but it gives us the determinant and the inverse almost for free.

4.3. Regularized Discriminant Analysis

We can shrink the covariance matrices towards their average:

$$\hat{\boldsymbol{\Sigma}}_k(\alpha) = \alpha\hat{\boldsymbol{\Sigma}}_k + (1-\alpha)\hat{\boldsymbol{\Sigma}}. \tag{17}$$

Of course, α has to be chosen, perhaps by cross-validation.

Similarly, we could shrink $\hat{\boldsymbol{\Sigma}}$ towards the scalar covariance matrix $\hat{\sigma}^2\mathbf{I}$:

$$\hat{\boldsymbol{\Sigma}}(\gamma) = \gamma\hat{\boldsymbol{\Sigma}} + (1-\gamma)\hat{\sigma}^2\mathbf{I}. \tag{18}$$

Combining (17) and (18), we could use

$$\hat{\boldsymbol{\Sigma}}_k(\alpha,\gamma) = \alpha\hat{\boldsymbol{\Sigma}}_k + (1-\alpha)\big(\gamma\hat{\boldsymbol{\Sigma}} + (1-\gamma)\hat{\sigma}^2\mathbf{I}\big), \tag{19}$$

which has two tuning parameters to specify.

In Section 4.3.3, ESL goes on to discuss reduced-rank LDA, which is closely related to LIML and to Johansen's approach to cointegration.

4.4. Logistic Regression

The log of the odds between any two classes is assumed to be linear in $\boldsymbol{x}$. This implies that

$$\Pr(G = k \,|\, \boldsymbol{x}) = p_k(\boldsymbol{x}, \boldsymbol{\theta}) = \frac{\exp(\beta_{k0} + \boldsymbol{x}^\top \boldsymbol{\beta}_k)}{1 + \sum_{\ell=1}^{K-1} \exp(\beta_{\ell 0} + \boldsymbol{x}^\top \boldsymbol{\beta}_\ell)}, \tag{20}$$

where $\boldsymbol{\theta}$ contains all of the parameters. There are $(K-1) * (p+1)$ of these.

ESL discusses ML estimation of the **logit model** ($K = 2$) in Section 4.4.1. What they have there, and later in Section 4.4.3 on inference, is closely related to the material in Sections 11.2 and 11.3 of ETM.

They do not discuss estimation of the multinomial logit (multilogit) model except in Exercise 4.4.

For binary logit, the probability of class 1 is

$$p(\boldsymbol{x}, \boldsymbol{\theta}) = \frac{\exp(\beta_0 + \boldsymbol{x}_i^\top \boldsymbol{\beta})}{1 + \exp(\beta_0 + \boldsymbol{x}_i^\top \boldsymbol{\beta})} = \frac{1}{1 + \exp(-\beta_0 - \boldsymbol{x}_i^\top \boldsymbol{\beta})}. \tag{21}$$

Thus the contributions to the loglikelihood are

$$\beta_0 + \boldsymbol{x}_i^\top \boldsymbol{\beta} - \log\bigl(1 + \exp(\beta_0 + \boldsymbol{x}_i^\top \boldsymbol{\beta})\bigr) \quad \text{if} \quad y_i = 1 \tag{22}$$

and

$$-\log\bigl(1 + \exp(\beta_0 + \boldsymbol{x}_i^\top \boldsymbol{\beta})\bigr) \quad \text{if} \quad y_i = 0. \tag{23}$$

The sum of these contributions over all observations is the loglikelihood function:

$$\ell(\boldsymbol{\beta}) = \sum_{i=1}^{N} \Bigl(y_i(\beta_0 + \boldsymbol{x}_i^\top \boldsymbol{\beta}) - \log\bigl(1 + \exp(\beta_0 + \boldsymbol{x}_i^\top \boldsymbol{\beta})\bigr)\Bigr). \tag{24}$$

To maximize $\ell(\boldsymbol{\beta})$, we differentiate with respect to β_0 and each element of $\boldsymbol{\beta}$ and set the derivatives to 0.

The first-order condition for β_0 is interesting:

$$\sum_{i=1}^{N} y_i - \sum_{i=1}^{N} \frac{\exp(\beta_0 + \boldsymbol{x}_i^\top \boldsymbol{\beta})}{1 + \exp(\beta_0 + \boldsymbol{x}_i^\top \boldsymbol{\beta})} = 0. \tag{25}$$

So the sum of the y_i must be equal to the sum of the probabilities that $y = 1$. Thus, at the ML estimates, the expected number of 1s must equal the actual number.

This is similar to the condition for OLS that the mean of the regressand must equal the mean of the fitted values.

The loglikelihood (24) can be maximized by a quasi-Newton method that is equivalent to iteratively reweighted least squares. See ESL, p. 121 or ETM pp. 455-456.

4.5. Regularized Logistic Regression

Of course, we can penalize the loglikelihood function for (multinomial) logit, using either an L_1 or L_2 penalty, or perhaps both in the fashion of the elastic-net penalty.

ESL only discusses the L_1 (lasso) case.

For the logit/lasso case, instead of maximizing

$$\sum_{i=1}^{N}\Big(y_i(\beta_0 + \boldsymbol{x}_i^\top\boldsymbol{\beta}) - \log\big(1 + \exp(\beta_0 + \boldsymbol{x}_i^\top\boldsymbol{\beta})\big)\Big), \tag{26}$$

we would maximize

$$\frac{1}{N}\sum_{i=1}^{N}\Big(y_i(\beta_0+\boldsymbol{x}_i^\top\boldsymbol{\beta})-\log\big(1+\exp(\beta_0+\boldsymbol{x}_i^\top\boldsymbol{\beta})\big)\Big)-\lambda\sum_{j=1}^{p}|\beta_j|. \tag{27}$$

The factor of $1/N$ is there to make λ not depend on N. Maximizing (27) apparently requires nonlinear programming, but specialized procedures based on coordinate descent are much faster.

For logit/ridge, we would replace the penalty term in (27) by

$$-\lambda\sum_{j=1}^{p}\beta_j^2=-\lambda||\boldsymbol{\beta}||^2=-\lambda\boldsymbol{\beta}^\top\boldsymbol{\beta}. \tag{28}$$

I found a paper (le Cessie and van Houwelingen, JRSS C, 1992, 1531 cites) that discusses this in detail. Estimation can be done using quasi-Newton methods, and cross-validation can be used to choose λ.

Of course, we need to standardize the predictors before we use any sort of penalty.

Although statisticians and econometricians typically use 0 and 1 as the values of y_i for classification problems with two classes, the machine learning community typically uses -1 and $+1$.

With this convention, the negative of the loglikelihood (24) is replaced by

$$\ell(\boldsymbol{\beta}) = \sum_{i=1}^{N} \log\left(1 + \exp(-y_i(\beta_0 + \boldsymbol{x}_i^\top \boldsymbol{\beta})\right). \tag{29}$$

Define $f(\boldsymbol{x}_i)$ as $\beta_0 + \boldsymbol{x}_i^\top \boldsymbol{\beta}$. Then $y_i f(\boldsymbol{x}_i)$ is called the **margin**. When the margin is positive/negative, the classification is correct/ incorrect.

4.6. Logistic Regression and LDA

Recall from (6) that the log of the odds between classes k and K for LDA is

$$\begin{aligned} &\log \frac{\pi_k}{\pi_K} - \frac{1}{2}(\boldsymbol{\mu}_k - \boldsymbol{\mu}_K)^\top \boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_k - \boldsymbol{\mu}_K) + \boldsymbol{x}^\top \boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_k - \boldsymbol{\mu}_K) \\ &= \alpha_{k0} + \boldsymbol{x}^\top \boldsymbol{\alpha}_k. \end{aligned} \tag{30}$$

For logistic regression, the log of the same odds is

$$\beta_{k0} + \boldsymbol{x}^\top \boldsymbol{\beta}_k. \tag{31}$$

Except for the names of the coefficients, (30) and (31) look identical!

However, the coefficients are estimated differently. When we estimate LDA and logit models, we maximize different things.

The joint density of $\boldsymbol{x}$ and G is

$$\Pr(\boldsymbol{x}, G = k) = \Pr(\boldsymbol{x}) \Pr(G = k|\boldsymbol{x}). \tag{32}$$

LDA maximizes the loglikelihood function

$$\sum_{i=1}^{N} \sum_{k=1}^{K} \log \Pr(\boldsymbol{x}_i, G = k) = \sum_{i=1}^{N} \sum_{k=1}^{K} \big(\log \phi(\boldsymbol{x}_i; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}) + \log \pi_k \big), \tag{33}$$

where $\phi(\cdot)$ is the multivariate normal density. This is based on the joint density (32) of G and $\boldsymbol{x}$.

Implicitly, the joint density that appears in (33) depends on the marginal density

$$\Pr(\boldsymbol{X}) = \sum_{k=1}^{K} \pi_k \phi(\boldsymbol{x}_i; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}), \tag{34}$$

which we obtain by summing over all the classes.

In contrast, multinomial logit maximizes the loglikelihood function

$$\sum_{i=1}^{N} \sum_{k=1}^{K} \log \Pr(G = k | \boldsymbol{x}_i), \tag{35}$$

which is based on the probability of $G = k$ conditional on $\boldsymbol{x}$. It does not depend on the marginal density (34).

Because LDA uses more information, it should be more efficient than multilogit. ESL cites a paper by Efron that says the efficiency loss is about 30%.

The loss could be much greater if we have lots of unlabelled observations. They can be used to estimate $\boldsymbol{\Sigma}$ even though we don't know what class they belong to.

However, the additional information comes from the assumption that the $\boldsymbol{x}$ are multivariate normal, which is a very strong assumption. Since (35) conditions on the $\boldsymbol{x}_i$, the multilogit model makes no assumption about how they are generated.

Moreover, the assumptions of LDA make no sense if any of the regressors is categorical. Thus multilogit is safer and more widely applicable.

4.7. Structured Local Regression Models

Unless p is very small, we need to impose structure on the model.

One approach is to use a **structured kernel** such as

$$K_{\lambda \boldsymbol{A}}(\boldsymbol{x}_0, \boldsymbol{x}) = D\left(\frac{(\boldsymbol{x} - \boldsymbol{x}_0)^\top \boldsymbol{A}(\boldsymbol{x} - \boldsymbol{x}_0)}{\lambda}\right), \tag{36}$$

where $\boldsymbol{A}$ is a positive definite matrix that gives more or less weight to certain directions.

ESL prefer to use **structured regression functions** such as

$$f(\boldsymbol{x}) = \alpha + \sum_{j=1}^{p} g_j(x_j) + \sum_{k<\ell} g_{k\ell}(x_k, x_\ell) + \dots \tag{37}$$

These are generalizations of the partially linear regression model. Typically, there cannot be too many higher-order terms. For **additive models**, there are just the p functions $g_j(x_j)$.

Instead of a nonlinear function for one variable and a linear model for all the others, (37) has many one-dimensional and two-dimensional nonlinear models to estimate.

This can be done iteratively. Consider the additive case. If we centre the data, and all the $g_j(x_j)$ except $g_k(x_k)$ are assumed known, we can estimate an additive model by repeatedly running the local regression

$$y - \sum_{j \neq k} g_j(x_j) = g_k(x_k) + \text{resid.} \tag{38}$$

We cycle through j from 1 to p until convergence. We may have to estimate a lot of local regressions, but each one is just one-dimensional.

Another type of model is the **varying coefficient model**. Let z denote x_p and define $q \equiv p - 1$. Then consider the model

$$f(\boldsymbol{x}) = \beta_0(z) + \beta_1(z)x_1 + \ldots + \beta_q(z)x_q, \tag{39}$$

where there are now p nonlinear functions to estimate. Conditional on them, we simply have a linear regression model.

It can be fitted by locally weighted least squares. We minimize

$$\sum_{i=1}^{N} K_\lambda(z_0, z_i)\big(y_i - \boldsymbol{x}_i^\top \boldsymbol{\beta}(z_0)\big)^2 \tag{40}$$

with respect to the vector $\boldsymbol{\beta}(z_0)$ for each value of z_0.

4.8. Local Likelihood

Any parametric model can be converted to a local one by using weights that vary across observations according to the value of $\boldsymbol{x}$.

In particular, it is easy to turn globally linear models into locally linear ones.

Suppose the model has a loglikelihood function

$$\ell(\boldsymbol{\beta}) = \sum_{i=1}^{N} \ell(y_i, \boldsymbol{x}_i^\top \boldsymbol{\beta}). \tag{41}$$

An obvious example is a logit or probit model. Then we can estimate a locally linear version by maximizing

$$\ell\big(\boldsymbol{\beta}(\boldsymbol{x}_0)\big) = \sum_{i=1}^{N} K_\lambda(\boldsymbol{x}_0, \boldsymbol{x}_i)\ell\big(y_i, \boldsymbol{x}_i^\top \boldsymbol{\beta}(\boldsymbol{x}_0)\big). \tag{42}$$

Here we weight contributions to the loglikelihood instead of squared residuals. We could also estimate a model with varying coefficients by maximizing

$$\ell\big(\boldsymbol{\theta}(z_0)\big) = \sum_{i=1}^{N} K_\lambda(z_0, z_i)\ell\big(y_i, \boldsymbol{x}_i^\top \boldsymbol{\theta}(z_0)\big) \tag{43}$$

with respect to the vector $\boldsymbol{\theta}(z_0)$ for each value of z_0; compare (40).

Consider the multilogit model with J responses, where

$$\Pr(G = j \mid \boldsymbol{x}) = \frac{\exp(\beta_{j0} + \boldsymbol{x}^\top \boldsymbol{\beta}_j)}{1 + \sum_{k=1}^{J-1} \exp(\beta_{k0} + \boldsymbol{x}^\top \boldsymbol{\beta}_k)}, \tag{44}$$

where $\beta_{J0} = 0$ and $\boldsymbol{\beta}_J = \mathbf{0}$.

The local loglikelihood for this model is

$$\sum_{i=1}^{N} K_\lambda(\boldsymbol{x}_0, \boldsymbol{x}_i)\Bigg(\beta_{g_i 0}(\boldsymbol{x}_0) + (\boldsymbol{x}_i - \boldsymbol{x}_0)^\top \boldsymbol{\beta}_{g_i}(\boldsymbol{x}_0) - \log\Bigg(1 + \sum_{k=1}^{J-1} \exp\left(\beta_{k0}(\boldsymbol{x}_0) + (\boldsymbol{x}_i - \boldsymbol{x}_0)^\top \boldsymbol{\beta}_{g_i}(\boldsymbol{x}_0)\right)\Bigg)\Bigg). \tag{45}$$

Because the regressions are centred at $\boldsymbol{x}_0$, the posterior probabilities at $\boldsymbol{x}_0$ are simply

$$\widehat{\Pr}(G = j \,|\, \boldsymbol{x}_0) = \frac{\exp\big(\hat{\beta}_{j0}(\boldsymbol{x}_0)\big)}{1 + \sum_{k=1}^{J-1} \exp\big(\hat{\beta}_{k0}(\boldsymbol{x}_0)\big)}. \tag{46}$$

Since $\widehat{\Pr}(G = j \,|\, \boldsymbol{x}_0)$ just depends on $\boldsymbol{x}_0$ and the coefficients $\hat{\beta}_{j0}$, $j = 1, J-1$, we can calculate its standard error using the delta method.

This model can be used for classification in low dimensions.

4.9. Kernel Estimation and Classification

This subsection is based on Sections 6.6 and 6.8 of ESL.

The Gaussian kernel, in ESL's notation, is

$$\hat{f}_X(x) = \frac{1}{N}\sum_{i=1}^{N}\phi_\lambda(x_i - x), \tag{47}$$

where $\phi_\lambda(\cdot)$ is the normal density with mean 0 and variance λ^2. The factor of $1/\lambda$ does not explicitly appear in (47) because it is part of ϕ_λ.

In the multivariate case, (47) generalizes to

$$\hat{f}_{\boldsymbol{X}}(\boldsymbol{x}) = \frac{1}{(2\lambda^2\pi)^{p/2}}\sum_{i=1}^{N}\exp\Big(-\tfrac{1}{2}\big(||\boldsymbol{x}_i - \boldsymbol{x}||^2/\lambda\Big). \tag{48}$$

Of course, the **curse of dimensionality** implies that we are likely to get very poor estimates when p is large and N is not extremely large.

We can use estimates like (48) for classification.

Suppose there are K classes. Then

$$\widehat{\Pr}(G = k \mid \boldsymbol{x}) = \frac{\hat{\pi}_k \hat{f}_k(\boldsymbol{x})}{\sum_{\ell=1}^{K} \hat{\pi}_\ell \hat{f}_\ell(\boldsymbol{x})}. \tag{49}$$

This is just the empirical counterpart of (3). In most cases, we use the sample proportions to estimate the $\hat{\pi}_k$, as in (8).

ESL note that, when classification is the goal, it is only points near the boundaries that we are interested in. We want to estimate the posterior probabilities accurately in those regions.

The **naive Bayes** estimator makes the extremely strong assumption that the joint density of the components of $\boldsymbol{x}$ is the product of the marginal densities:

$$f_k(\boldsymbol{x}) = \prod_{j=1}^{p} f_{kj}(x_j), \quad k = 1, \ldots, K. \tag{50}$$

For any $\boldsymbol{x}$ that interests us, we simply obtain p kernel density estimates, $\hat{f}_{kj}$, and use their product to estimate $\hat{f}_k(\boldsymbol{x})$.

If unrestricted kernel density estimates based on (48) are too noisy and the naive Bayes estimator is too biased, another possibility is to estimate **normal mixture models**.

In the multivariate case, a normal mixture has the form

$$f(\boldsymbol{x}) = \sum_{m=1}^{M} \alpha_m \phi(\boldsymbol{x}; \boldsymbol{\mu}_m, \boldsymbol{\Sigma}_m), \tag{51}$$

where the **mixing proportions** α_m sum to unity.

We can estimate a model like (51) for each class, and these lead to flexible models for $\Pr(G \,|\, \boldsymbol{x})$.

As previously discussed, we can also use kNN methods for classification. Simply classify $\boldsymbol{x}$ as belonging to whatever class is most common among the k nearest neighbours to $\boldsymbol{x}$.

4.10. The Curse of Dimensionality

The performance of kernel estimation (and smoothing methods in general, including kNN) rapidly deteriorates as p increases.

The fundamental problem is that the volume of a mathematical space increases exponentially with the number of dimensions.

The unit interval is a 1-dimensional hypercube. If we take 100 evenly-spaced points on a grid, the distance between them will be only $10^{-2} = 0.01$.

Any point we pick at random will be "close" to several of the points on the grid.

Suppose there are two dimensions instead of one. To get the same distance between points on the grid (in the direction of each axis), we need $100^2 = 10,000$ points.

Moreover, the average Euclidean distance between a point and its nearest neighbours will be greater than 0.01, so that a randomly chosen point in the hypercube (a square in this case) will be further away from the nearest point on the grid.

For three dimensions, we need $100^3 = 1,000,000$ points to achieve a distance of 0.01 in the direction of each axis, and the average Euclidean distance between a point and its nearest neighbours will be even greater.

Thus, unless the sample size increases very rapidly as p increases, the data become much "sparser" as p increases. Any estimator based on local averages will have fewer and fewer nearby points to average over as p increases.

When predictors take on discrete values, we get the same sort of problem. Suppose that each predictor is binary. Then the number of possible sets of values is 2^p.

For large p, this will be much larger than N, so many possible sets of predictor values will never appear in any given sample.

A machine-learning algorithm that does not impose strong assumptions about functional form cannot make reliable predictions about events that are not "close" to ones observed in the sample.

This is one reason why high-dimensional methods, such as lasso, ridge, and elastic net, do make strong assumptions about functional form.

We can handle problems with large p, or we can allow the relationship between inputs and outputs to be very general, but we cannot do both.